This is the author's version (manuscript) of the work that was accepted for publication in AIAA SCITECH 2024 Forum.
The Version of Record is available online at https://arc.aiaa.org/doi/10.2514/6.2024-0383 .

Integration of large data based on HDF5 in a collaborative and multidisciplinary design environment, Part B: Application to Structural Mechanics of Gas Turbine Engines

O. Kunc*
German Aerospace Center (DLR), Stuttgart, Germany, D-70569

M. Bröcker†
German Aerospace Center (DLR), Cologne, Germany, D-51147

We present a deep integration of structural 3D Finite Element data into the central data model of a collaborative aircraft engine design platform. This is a major step for the transition from preliminary design to detailed design in a consistent way on the same platform. Template based workflows leverage the directly available large data and ease the execution of CSM and interdisciplinary processes.

I. Nomenclature

ADP	=	Aerodynamic Design Point
CFD	=	Computational Fluid Dynamics
CGNS	=	CFD General Notation System
CSM	=	Computational Structural Mechanics
DLR	=	German Aerospace Center
FE(M)	=	Finite Element (Method)
FSI	=	Fluid Structure Interaction
GTlab	=	Gas Turbine Laboratory
HDF5	=	Hierarchical Data Format
I/O	=	input and output
SMM	=	Structural Mechanics Module
SWMR	=	Single Write Multiple Read

II. Introduction

Collaboration in engineering across multiple disciplines is becoming generally more important in order to achieve further technical improvements. This is especially true for air breathing gas turbine engines, which were developed and perfected for approximately 90 years. The collaboration and development platform Gas Turbine Laboratory (GTlab) [1, 2] by the German Aerospace Center (DLR) streamlines legacy methods and enables novel approaches in order to advance this field of propulsion technologies. One of its key achievements was the introduction of a central data model. This enables a standardized parametrization of conceptual and preliminary engine design.

Whereas the original central data model serves its purpose very well for lower fidelity designs, extensions towards higher levels of detail are now being introduced. One of many steps in this direction is to embed 3D simulation data and processes, such that parts of the data may be referenced, extended, re-used by subsequent processes, and shared with collaborators. As simulation data quickly grow in size with increasing level of engineering detail, the Hierarchical Data Format (HDF5) is a promising binary storage format for this application. The HDF5 plugin for GTlab was introduced in part A of this publication [3]. By means of this plugin, HDF5 data may be seamlessly and flexibly integrated into the GTlab data model. The particular realization and utilization of features provided by the HDF5 plugin is to be carried out by domain-specific module developers.

*Research Associate, Institute of Structures and Design, Pfaffenwaldring 38–40, 70569 Stuttgart, Germany, oliver.kunc@dlr.de
†Research Associate, Institute of Propulsion Technology, Linder Hoehe, 51147 Cologne, Germany, marius.broecker@dlr.de

This work presents the application of the HDF5 plugin to simulation data with an emphasis on – but not limited to – Computational Structural Mechanics (CSM). The aim is to seamlessly integrate CSM data into the central data model and pertinent workflows.

The outline of this paper follows the chronology of our work on this topic: Section III briefly introduces GTlab as a foundation and its novel HDF5 plugin. In Section IV a suitable standard for the storage of simulation data in the form of HDF5 files is summarized. The main Section V presents detailed use cases from CSM and related fields. Section VI summarizes both the achievements of the current work and what is yet to be done.

III. GTlab and its HDF5 plugin

A. Framework GTlab

The Gas Turbine Laboratory framework was introduced in [1, 2]. This in-house development by DLR provides a software platform for gas turbine development. Its main focus is on integration and collaboration of different disciplines. Without going into much detail, the gist of it is now recalled in order to provide enough context for the presented work.

1. Core modules

GTlab is a modular software. In its current version 2.0, it consists of several core libraries that provide basic functionalities. Multiple specialized plugins extend the functionalities of the core framework. For instance, the core data processor loads, (de-)serializes, and tracks changes of data objects as they are modified by the user or by process elements. The latter are registered data objects themselves, meaning the software provides means to handle changes in process configurations. Another part of the GTlab framework is the GUI library (Graphical User Interface) providing graphical functionalities. Whereas GTlab is first and foremost a graphical application, it also provides a console interface.

These and many other functionalities are actually rather general and by no means restricted to gas turbine development. Applications in unrelated fields of collaborative engineering are thinkable, but no such efforts have been made yet. An open source release of this core framework is anticipated in the coming months.

2. Structural Mechanics Module (SMM)

The SMM currently provides a collection of standard processes for structural mechanics of turbo components. It is being functionally extended with the goal of providing automated CSM capabilities directly to users of all disciplines, wherever such expertise is of interest. Shorter iteration times across all stakeholders are anticipated by distributing such template based workflows, including local back-end processing and a user-oriented front-end. The inputs to the SMM consist of geometries and performance data from the central data model. Examples of SMM workflows will be presented in later sections.

3. Central data model

The central data model, as detailed in [1, 2], is tailored to the design of turbo components. It is central in the sense that, by default, all processes use it for both input and output. Therefore, any changes made by a process can be registered (reliable output format) and are always compatible with subsequent processes (reliable input format). This paradigm provides a supportive structure for both users and developers across disciplines and organizations. However, processes may be augmented by arbitrary operations on the file system. This ensures sufficient flexibility in cases that processes require data that are not part of the central data model.

Geometries of components on different hierarchy levels, such as compressors, combustors, turbines, blades, disks, ducts, etc. are parametrically described within the central data model. Higher-detail features such as sealings, blade platforms, shrouds, and others are being developed.

In GTlab, the central data model is represented in form of a tree hierarchy. Particular instances of aircraft engine designs are organized as projects. Such a project layout is sketched in Fig. 1.

Apart from objects of the central data model, tabular data can be stored in the corresponding tree of a data record. This is intended to keep results of performance calculations, results of low-fidelity simulations or other supplementary, small datasets readily available. Large data, such as 3D CSM simulation models and results, are not intended to be stored in this way for multiple reasons, cf. [3].

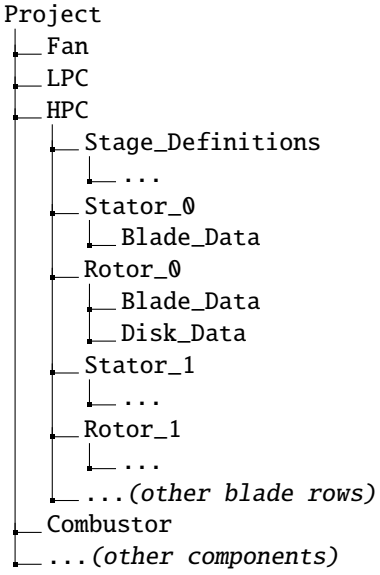

Fig. 1 Sketch of a data record based on the central data model of GTlab. All major components can be hierarchically defined.

The main reason for this is the methodology underlying the interaction of processes and data: upon execution, all data of the central data model are hashed. After de-serialization, all data operations take place in a "sandbox" memory area where modifications do not affect the original input data. Hashing of this working data is repeated upon process completion. State changes of data are registered by means of comparing hashes of inputs and outputs. Only modifications are stored from the "sandboxed" area to disk memory. This way, it is ensured that only successful processes modify the central data hierarchy. Furthermore, undo and redo operations are enabled by thereby.

Whereas this has proven very fruitful for parametric model data and modest amounts of tabular data, repeated hashing and complete de-serialization of large simulation data would quickly lead to excessive computational costs. Large simulation data, that are quantitatively but also qualitatively different from what is covered by the central data model require a fundamentally different approach to storage and handling. This is the purpose of the HDF5 plugin.

B. HDF5 plugin

The framework GTlab implements the central data model approach as depicted earlier. This approach address the challenges faced in collaborative design. It allows the coupling of various heterogeneous tools and data of different complexities. Further, the standardized exchange of data is facilitated. Additionally, the data provenance approach ensures data consistency throughout the design process [4]. However, high-fidelity processes and simulations (like CSM) pose challenges to the aforementioned approaches. Such processes are executed outside the collaborative architecture and often produce large data, frequently stored in binary formats. In the design environment of aircraft engines at DLR the data format HDF5 gained increased adoption as a common output format.

As a consequence, external processes and high-fidelity simulations can be considered as black-box transformations. The data produced is, first and foremost, opaque to the aforementioned approaches. A single HDF5 file may contain numerous datasets, structured in an internal hierarchy. However, to access the individual datasets, specialized tools are necessary. In the context of GTlab, this is facilitated by the HDF5 plugin, as depicted in part A [3]. It enables the indexing of an HDF5 file, making individual datasets referenceable and trackable. The integration does not require any changes to the core of the framework. As a result, the advantages of the central data model, which is essential for collaborative design, can be maintained. Additional graphical tools also provide a seamless and lightweight integration from a user's point of view. Therefore, by integrating large data based on HDF5 into the collaborative framework of GTlab, processes like CSM become integrative as well.

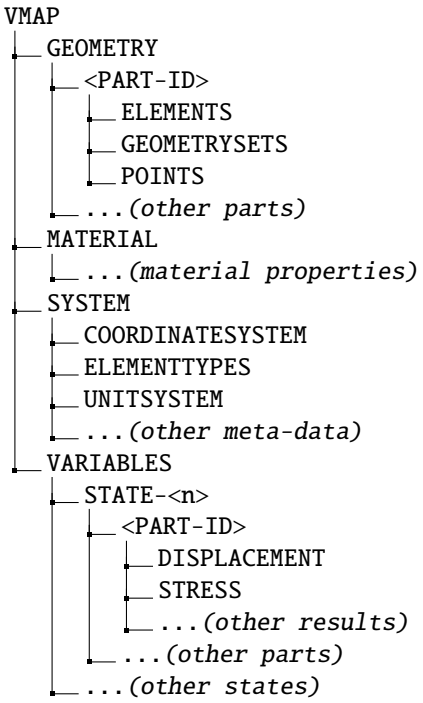

Fig. 2 Basic structure of an HDF5 file complying with the VMAP standard.

IV. The VMAP standard and its integration

The VMAP standard [5, 6] serves the purpose of efficient storage of Finite Element meshes and results with the goal of enabling process interoperability. Its current version V1.0.0 is defined in [7]. The essence of the standard shall be stated here in order to provide the reader with enough background information for the subsequent sections.

A. Brief introduction to VMAP

Figure 2 depicts the layout of a HDF5 file according to the VMAP standard. The main four (HDF5-)groups below the `/VMAP` level are now described as far as necessary for the current purpose.

The `GEOMETRY` group contains all FE model entities that are used for the description and reference of geometries and subsets thereof. Nodes, elements, and subsets of nodes, elements, and faces are stored inside this group. Notably, a `PART` sub-hierarchy sorts these entities with respect to parts. In contrast to some legacy "flat" model descriptions, descriptions based on parts allow for independent numbering inside each part and ease the assembly of multiple parts to form a larger component.

The `MATERIAL` group stores certain material properties, such as the Young's modulus, Poisson ratio, density, etc. Thermal dependencies may be stored in tabular form and certain meta-data may also be included, such as the source of the specific material, or attributes specific to a certain FE software.

The `SYSTEM` group contains diverse data complementary to the other groups, such as coordinate systems, physical unit systems, and FE integration schemes.

The `VARIABLES` group stores the simulation results. Its first sub-hierarchy consists of a list of `STATE`s in order to accommodate results from different simulation times. Within each `STATE` group, the `PART` hierarchy of the `GEOMETRY` group may be found, in its entirety or partially, depending on the association of the stored results. Therefore, the actual result values from the next-level sub-groups (such as `DISPLACEMENT`, `TEMPERATURE`, etc.) are directly connected to their respective domain. Moreover, it is possible to associate the results with subsets, e.g. with certain surfaces.

The VMAP Standards Community, cf. [5], is currently working on an extension of the VMAP standard in order to include boundary conditions, initial conditions, and loads. It is anticipated that the next versions of VMAP will – for many use cases – enable complete storage of model inputs and results, reproducible and interchangeable with a wide range of FE simulation software. There are converters between VMAP and more than 25 software products (some of which even provide native compatibility with VMAP) at the time of writing, cf. [7, "Tools"].

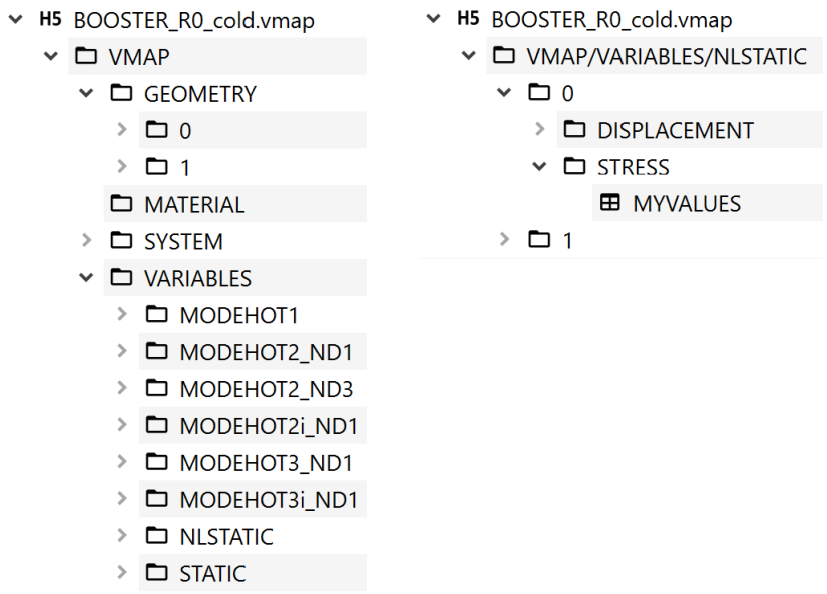

Fig. 3 Exemplary structure of a VMAP file as created by and used within the SMM. *Left*: representation of the entire file (standard). *Right*: shortcut to the `/VMAP/VARIABLES/NLSTATIC` result groups.

B. Integration of VMAP into the Central Data Model of GTlab

As described in section III.B, HDF5 files such as VMAP files are attached to corresponding data model objects. At the time of writing, VMAP files may be appended to the data node of each blade row, i.e. to each rotor and each stator section respectively. Such a file and its representation in the GTlab data tree is automatically created the first time a corresponding SMM process is run, usually a meshing task.

Figure 3, left, exemplifies the content of a VMAP file as used by processes provided by the SMM. The `PART` with ID `0` always represents the blade. In case of a rotor section, the `PART` labeled `1` represents the rotor disk.

At the time of writing, a slight deviation from the VMAP standard V1.0.0 is implemented by assigning the direct sub-groups of `VARIABLES` more meaningful names, such as `STATIC` and `MODE...`, in contrast to the standardized `STATE-X`. This does not significantly violate compliance with the standard. The future VMAP standard extension, as described in Section IV.A, is expected to incorporate the use cases provided by the SMM of GTlab.

Under certain circumstances, it may be convenient to focus on specific sub-hierarchies of a VMAP file. This is especially helpful when the overall structure contains many nodes, e.g. parts and results. Therefore, the feature of creating shortcuts to individual HDF5 nodes is highly appreciated, cf. [3, Section V.B]. An examples of this setting is depicted in Figure 3, right.

V. HDF5 use cases from a CSM perspective

A. Process automation

This section elaborates on use cases of the previously described HDF5 capabilities of GTlab. These use cases are sorted in tree chronological phases: the beginning (Subsection V.A.1), geometry and meshing (Subsection V.A.2), and simulation results (Subsection V.A.3). Advantages of HDF5 that apply to all use cases are listed (Subsection V.A.4).

The main focus lies on the relationship between SMM-provided CSM processes and HDF5 files in VMAP format. As the workflows provided by SMM are designed to inter-operate with processes of other GTlab modules, some interactions between CSM and Computational Fluid Dynamics (CFD) are also exemplified. Such interactions are common within the interdisciplinary field of Fluid Structure Interaction (FSI).

We consider the case of a fan or compressor blisk, i.e., a monolithic bladed rotor disk. The particular examples of the following subsections serve the purpose of substantiating the described processes and are not the main focus. All of the processes are example-agnostic templates provided by the SMM, i.e. they may be applied to the respective components of any gas turbine engine.

1. Phase: beginning

The input to the following process chain is the geometry of a single blade. It was previously designed by means of performance calculations and aerodynamic methods during the preliminary design phase. This loaded – or "hot" – geometry is described with respect to the Aerodynamic Design Point (ADP) and is assumed to be present within the central data model, cf. Figure 4.

No HDF5 file exists at this point in time. The data model solely contains the geometric modeling parameters of the blade and, if applicable, other results of prior, low-fidelity design methods.

Fig. 4 Exemplary compressor blade corresponding to the ADP. Displayed by the 3D viewer of GTlab.

2. Phase: geometry and meshing

Four sub-processes exist that concern the modification or the meshing of the geometry. They are now listed in one of multiple possible chronological orders. If a particular application scenario requires a different order, the modular nature of these GTlab process elements allows for an easy re-ordering. All four processes mainly read and/or write within the group `/VMAP/GEOMETRY`. However, secondary results might be stored to `/VMAP/VARIABLES` for later use.

Figure 5 contains a linear flowchart of these processes and their inputs and outputs. Figure 6 depicts the same processes in a data-centered flowchart with an actual data tree as available in GTlab. Figure 7 shows the result of the processes, an example mesh of a segment of a bladed rotor disk in cold state.

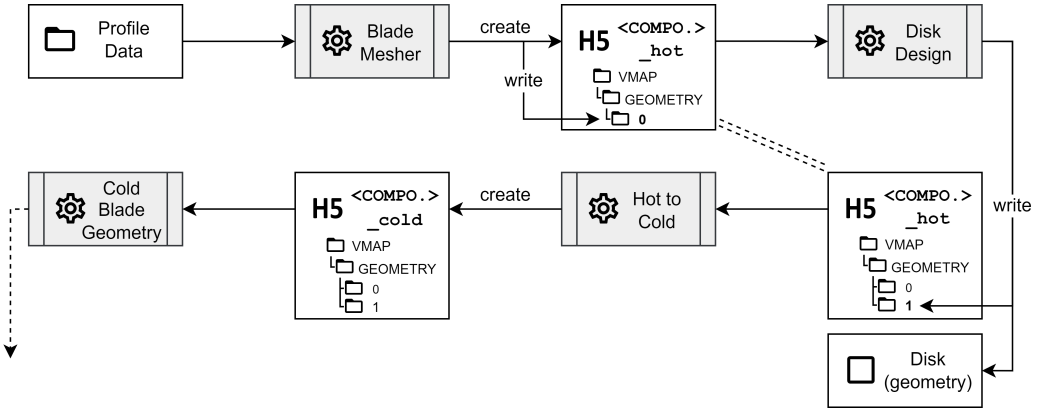

Fig. 5 Linear workflow chart of four exemplary processes. Two HDF5 files (H5 icon) in compliance with the VMAP standard are automatically created and interacted with.

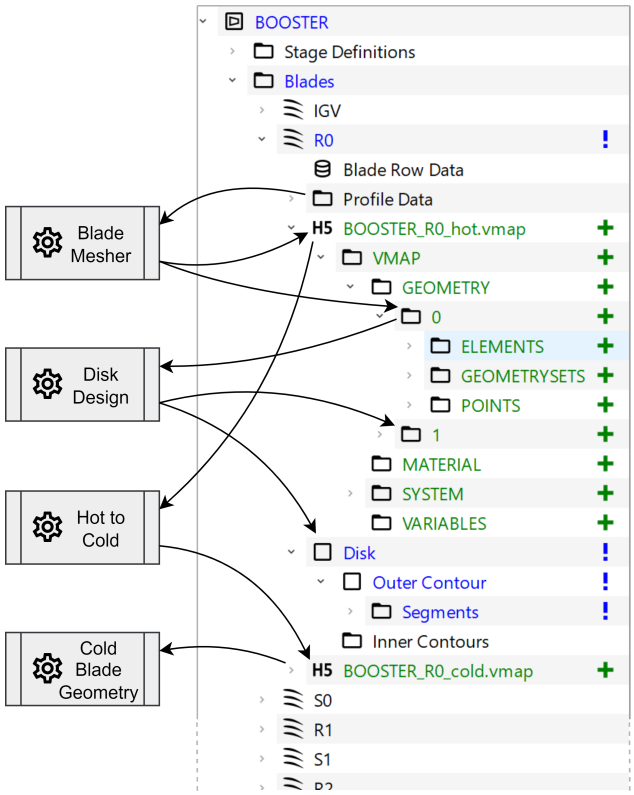

Fig. 6 The four meshing and geometry processes of Fig. 5 and their interactions with the central data model (hierarchy tree). Two HDF5 files (H5 icon) in VMAP format are attached to the data tree. These are stored in the local project folder by the names `BOOSTER_R0_hot/...cold.vmap` to indicate their relationship with the other data objects. The font color of the data objects indicate change of state: black represents no change, green (+) represents a newly created data object, blue represents change of the object itself (!) or its children (no symbol).

Use case 1: hot blade mesh, creation of HDF5 file

The hot blade geometry is meshed by means of a self-written algorithm. This algorithm directly operates on the geometric description within the data model and creates a structured mesh of linear hexahedrons. The interested reader is referred to [8] for more details in this regard.

The VMAP file `<COMPONENT>_hot.vmap` is created and appended to the data model of the rotor component. In this context, appending means the creation of a link to the file and indexing its contents, without reading any dataset. Usual filename prefixes in place of `<COMPONENT>` are `FAN_R0`, `LPC_R0`, `HPC_R1`, etc., depending on the names of the parent hierarchy and of the particular rotor component.

The mesh consists of nodes and elements which are stored as part `0`, i.e. to the group `/VMAP/GEOMETRY/0`. This group may optionally also contain the sub-group `GEOMETRYSETS` that can store sub-sets of nodes, elements, surfaces, etc. for later use.

Use case 2: disk geometry, mesh, and optimization

The central data model contains a collection of parameters that enable the description of a disk contour in the x-r-plane by means of splines and polygons. The entire description consists of roughly 10 parameters, each describing a particular segment of the contour. The transition from one segment to the other may be smooth or cornered.

After an initial disk contour is set via standard parameters, the disk is implicitly defined via rotation of the contour around the engine's x-axis. Thus an initial, rotationally symmetric disk geometry is created. The interface to the blade is guaranteed to be compatible to the blade root profile, given a user-defined tolerance.

Next, a segment of the disk is meshed. To this end, the two-dimensional parametric contour is meshed with unstructured linear quadrilaterals using the open source software Gmsh [9]. Just like its bounding contour this mesh

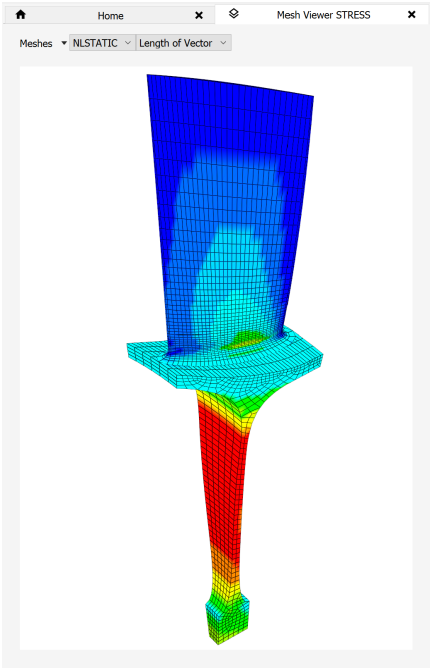

Fig. 7 Mesh of the cold version of Fig. 4 in blisk configuration, displayed by the GTMesh module reading VMAP.

is then circumferentially extruded. However, this extrusion is stopped after $2\pi/N_{\text{blades}}$ radians as only a single disk segment is required. The extrusion is subdivided regularly in order to generate a mesh of linear hexahedrons with equal circumferential extension.

An optimization of the disk geometry is also implemented in the SMM. This iterative procedure goes beyond the main scope of this section but should be mentioned for the sake of completeness. In each iteration, the definition of the disk contour is overwritten. The corresponding parameters of each iteration are stored in a separate entity of the data model which is only relevant to this particular sub-task. The disk mesh in `/VMAP/GEOMETRY/1` is also overwritten in each iteration. The goal of the optimization is mass minimization under certain restrictions of displacements and stresses. The results of the corresponding FE analyses of the last iteration may be kept in the group `/VMAP/VARIABLES/STATIC`. More details can be found in [8].

Use case 3: Hot-to-Cold Transform

This process performs an iterative hot-to-cold transformation of FE models, similar to [10, Section 3]. The main interest lies on the blade, i.e. part `0` of the VMAP file, but the disk, i.e. part `1`, may also be subject to this transformation. In any case, the resulting "cold coordinates" are stored to the group `/VMAP/GEOMETRY` of a new file `<COMPONENT>_cold.vmap`.

The mesh and the respective sub-entities are also written to this new file at the moment. It is planned to incorporate HDF5-references from `<COMPONENT>_cold.vmap` to `<COMPONENT>_hot.vmap` in order to avoid redundancies of the mesh connectivity and the sub-sets in the future.

Use case 4: cold blade geometry

The cold blade geometry is of interest for multiple subsequent processes such as detail design or manufacturing, all of which are beyond the scope of this work. Here, we restrict ourselves to noting that the contents of the mesh file `<COMPONENT>_cold.vmap` may be used for the determination of a cold blade geometry along the lines of [11]. Such a process is being implemented within the SMM. It is also anticipated to determine the cold equivalent of the BladeGen parameters. Then, the process chain sketched in Figure 5 would result in output of the same kind as the input, but with different numerical content.

3. Phase: simulation results

It is assumed that the geometry and meshing phase is completed. Now the primary interest is in simulation results, which are stored in the group `/VMAP/VARIABLES`. The following list is an excerpt of processes that are operational or are being developed at our institutions. For static and modal CSM analyses, the open source software CalculiX [12] is used. CFD simulations are performed using the software TRACE [13].

Use case 5: gap assessment and static stresses

A static simulation of the cold blisk is conducted with maximum rotational load and an optional averaged pressure load. If limit stresses are exceeded or clearances are out of bounds then the input geometry may need to be changed and re-assessed from an aerodynamic and/or performance perspective. In this case, go to the beginning, i.e. Section V.A.1. See Figure 8 for a workflow chart.

Use case 6: birdstrike simulation (fan only)

Birdstrike is a critical load case for fans. The involved kinetic energy is extreme and the possible consequences may be significant. It is therefore a dimensioning load case for the first rotating component. Automation of this process eases the assessment of this load case at a comparatively early stage, e.g. during advanced preliminary design, possibly avoiding design iterations involving higher levels of detail.

The cold meshes of both blade and disk are exported to text files that can be processed by an external LS-DYNA routine. This Python-based routine includes the duplication of the single blisk segment, the arrangement of the duplicates, setting of boundary conditions, etc. All necessary parameters are provided by the exporter via specific Python files.

After this pre-processing, two consecutive LS-DYNA simulations are performed: the static pre-impact situation and the dynamic impact itself. The former contains rotational loads (and possibly also the fluid pressure), whereas the latter additionally consideres the impact load.

Outputs of LS-DYNA are displayed after potential post-processing by this Python-based routine. The user is informed of the requested criteria, such as blade-off, clashing of neighboring blades, maximum plastic deformation, etc.

In case of a negative assessment, the input geometry may need to be changed and re-assessed from an aerodynamic and/or performance perspective. Go to the the beginning, i.e. Section V.A.1. See Figure 9 for a workflow chart and [8] for more details. It should be noted that the mesh creation methods described in Section V.A.2 can optionally optimize the mesh with respect to element quality criteria suitable for impact simulations by LS-DYNA, e.g. in order to maximize the temporal step size.

Use case 7: static FSI deformation (focus on CSM)

The goal of this workflow is to find an equilibrium state between the mechanics of a rotor blade and the thermodynamics of its surrounding fluid, given the rotational speed and certain conditions at the fluid inlet and outlet corresponding to an arbitrary operating point. Figure 10 sketches a simplified workflow that is currently being implemented.

The CFD software initially creates (and in subsequent iterations modifies) an HDF5-based CFD General Notation System file (CGNS). The relevant surface results are mapped onto the meshed blade structure by the Mapping tool, which is already integrated in the SMM. These mapped values are stored in `/VMAP/VARIABLES/BC_CFD` ("Boundary Conditions CFD").

The subsequent CSM process formulates appropriate FE boundary conditions based on these mapped values. The FE results are stored in `/VMAP/VARIABLES/NLSTATIC` ("Nonlinear static analysis"). Closing the circle, a Morphing tool which is not part of the SMM extrapolates the surface values of the displacements into the fluid volume, cf. [14]. These morphed grid coordinates of the fluid volume may be stored to the same CGNS file, stored to a new CGNS file, or used directly by the CFD software without dedicated file input and output (I/O). More details on this topic can be found, e.g., in [15, Section 3.3].

Iterations are performed until certain convergence criteria are met. This static result, i.e. the ensemble of conditions of both the fluid and the structure, may be the input to subsequent processes, such as the final item in this list of examples.

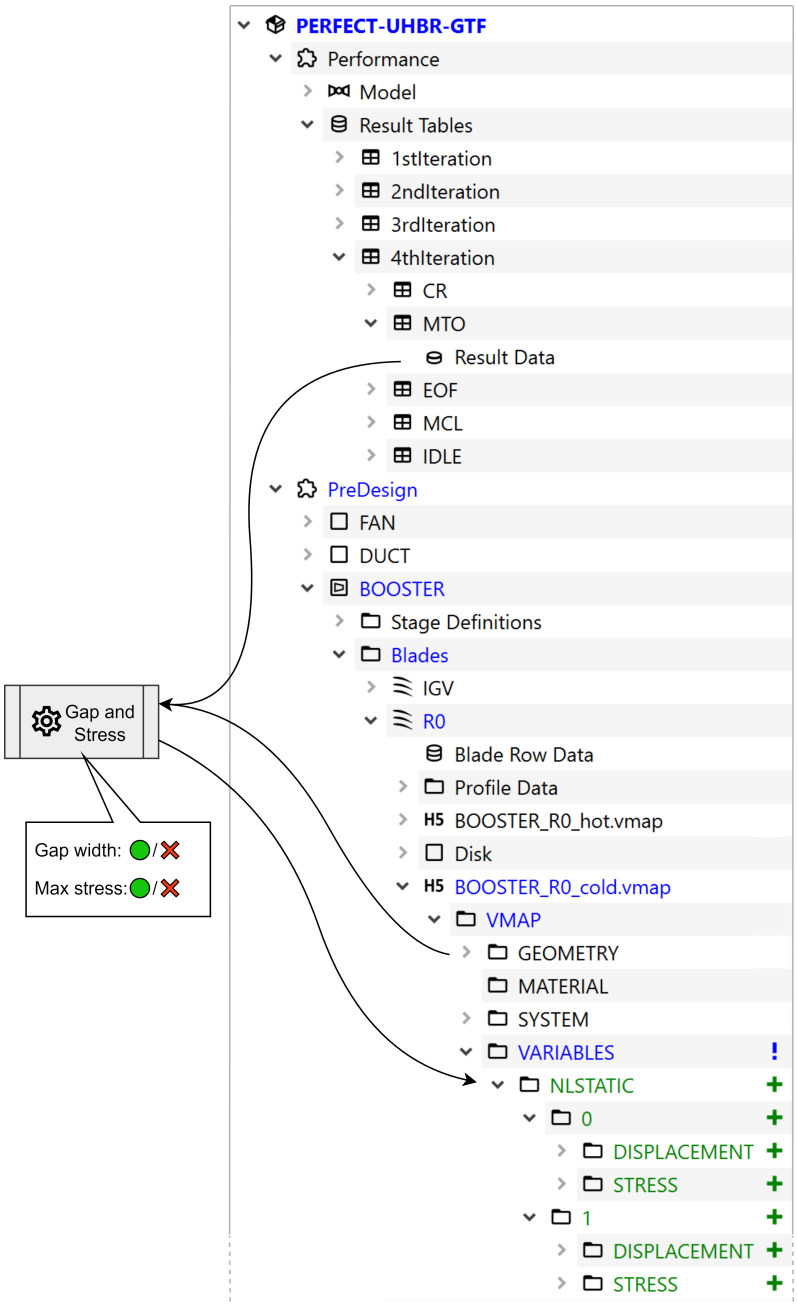

Fig. 8 Process of preliminary assessment of gap width (blade tip clearance) and maximum stress. The rotational velocity and an average pressure value are taken from results of the Performance module, in this example the values for take off at maximum weight (MTO). FE results are added to the existing VMAP file. A message is displayed to the user, stating the values of interest and their compliance with or violation of the safety criteria (green circle sign or red cross, respectively).

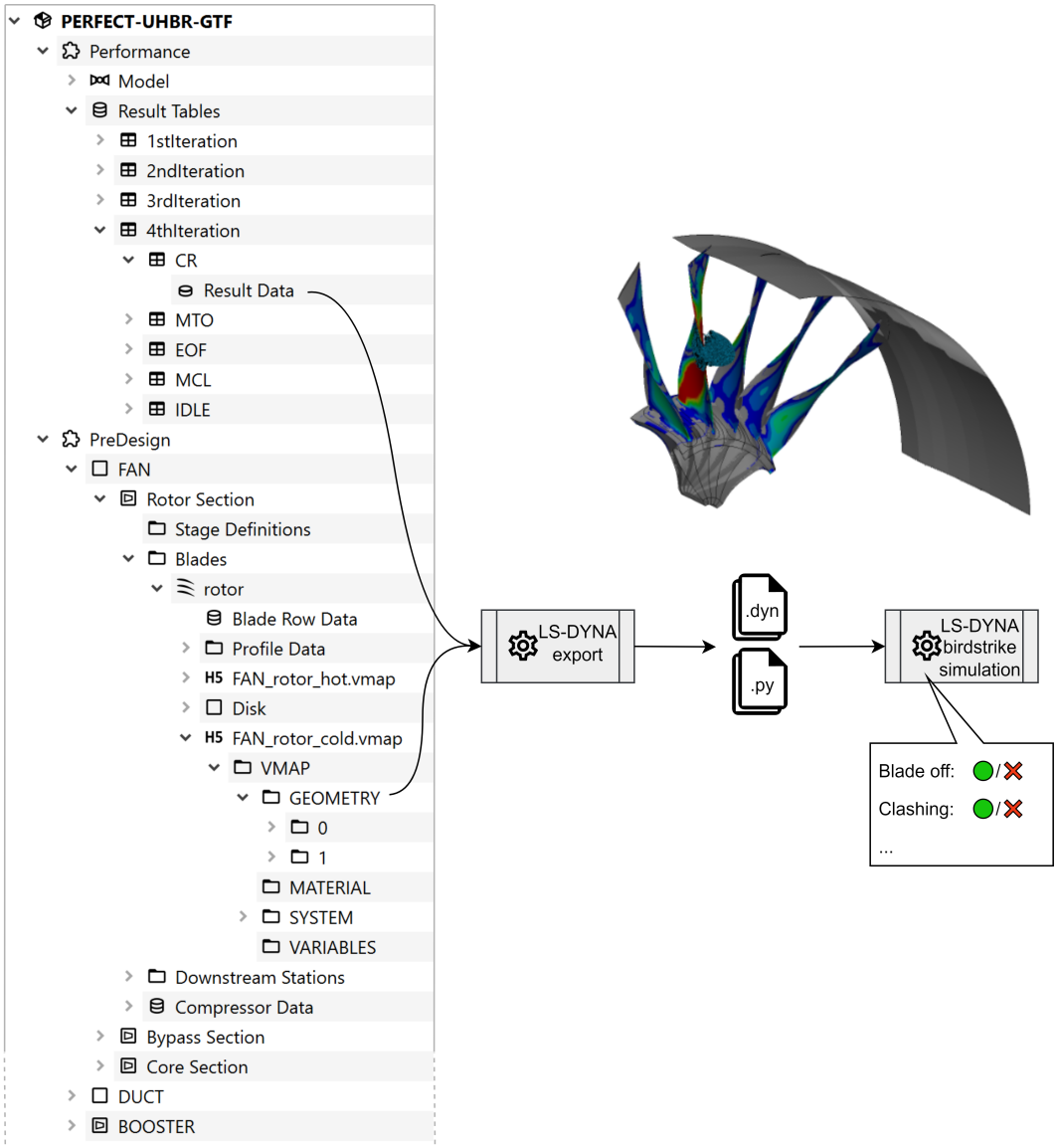

Fig. 9 Birdstrike workflow. Inputs are the rotational velocity (and optionally an averaged pressure value) from the Performance module, and the cold meshes of the blade and of the disk. The LS-DYNA export process creates dyna key-files and some Python script files that are specific to the subsequent external process. The latter pre-processes and uses said files for the execution of two LS-DYNA runs (static pre-load followed by impact). Post-processing is also conducted in this external process, and assessment results are made available to the user.

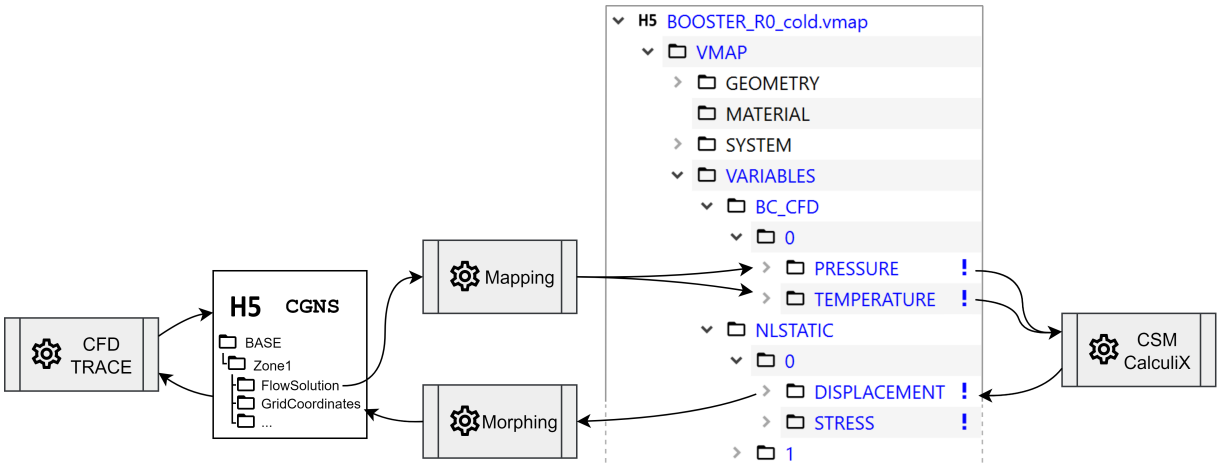

Fig. 10 Static FSI, simplified concept. The Mapping and the CSM CalculiX processes are integrated into the SMM. The Morphing and the CFD TRACE processes are external.

Use case 8: modal FSI (CSM part of Forced Response or Flutter Analysis)

This final example can only be touched very briefly in the context of this work. See Figure 11 for a rough simplification of the workflow and [15, 16] for more details on Forced Response and Flutter Analysis.

On the cold geometry, a static non-linear pre-load analysis is conducted. This accounts for at least the rotational load but also possibly for pressure loads. The resulting displacements and stresses are stored to `/VMAP/VARIABLES/NLSTAT_PRE_ND`. The choice of this nomenclature allows for the parallel storage of other static results, such as `NLSTATIC` from previous use cases.

Nodal diameter analyses are performed accounting for the elevated stiffness of the structure under this pre-load. A certain number of mode shapes are computed for each nodal diameter. Both real and imaginary results are obtained for most of these mode shapes. The results are conclusively stored to the groups `/VMAP/VARIABLES/MODEHOT`X`_NDY`, where $X = 1, 1i, 2, 2i, \ldots$ denotes the mode number and its nature (real or imaginary), and $Y = 0, 1, 2, \ldots$ denotes the nodal diameter. The stored results contain the displacements and the corresponding stresses and frequencies.

Note that the VMAP standard allows for complex numbers in a single dataset, i.e. the combined storage of real and imaginary values. At the current stage of our developments, storing the real and imaginary values separately appears slightly advantageous for subsequent processes that are yet to be implemented.

The relevant surface data are exported into a format that is suitable for subsequent CFD and aeroelastic processes. Additionally, mode shapes may be classified via the Modal Assurance Criterion [17], and Campbell diagrams may be created. After structural and aeroelastic assessment, the input geometry may need to be changed and re-assessed from an aerodynamic and/or performance perspective. In such case, go to the beginning V.A.1.

4. Advantages of HDF5-based simulation data workflows

All of the above mentioned use cases benefit from the usage of HDF5 instead of classical ASCII-encoded text files for the following four main reasons.

1) **Direct flow of data, no detour via ASCII files**

Classically, many in-house tools for special use cases are based on ASCII files. This often leads to the creation of temporary files that serve the mere purpose of transferring data output from one tool as input to the next tool. Even if the next tool may directly read the ASCII data of the previous tool, efforts for robust parsing need to be made and come at the expense of computational costs and longer development times. Changes in data structures, e.g., for the adaption to new use cases, often result in re-writes of large portions of code.

With HDF5, significant file I/O operations are avoided by bypassing writing, reading, and parsing intermediate ASCII files. The structured nature of HDF5 facilitates tool chains reading input data for one tool directly from the output data of the previous tools. The experience of the authors is that testing, maintaining, and adapting such HDF5-based process chains requires significantly less development effort than the ASCII-based alternatives.

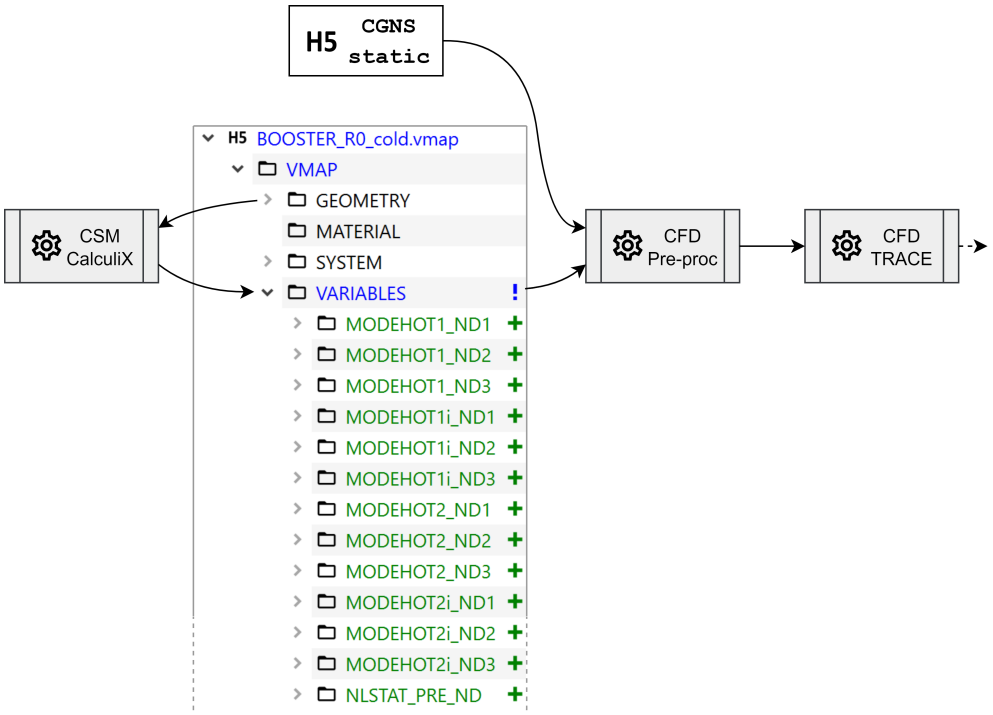

Fig. 11 Modal FSI, simplififed conncept. This one-way process builds on results of nodal diameter calculations (`ND`) and of the static FSI, cf. Figure 10. In the depicted example, three ND analyses (1–3) are conducted, computing the first two eigenmodes (1–2). These are called `HOT` beacause they are based on a non-linear static pre-load (`NLSTAT_PRE_ND`) of the cold geometry. The real and the imaginary (`i`) parts are stored separately. Each result group contains displacements and stresses of the blade and, if required, of the disk.

2) **Parallel data processing**
   HDF5 natively supports Single Write Multiple Read (SWMR) mode. This means that reading processes can be directly implemented in a parallel manner without the necessity for additional libraries or frameworks.
   For example, the FSI mapping process, cf. Fig. 10, may be split into parallel (quasi-)processes. Each of these reads CGNS and VMAP values of a certain subdomain, performs the computationally intense mapping and then writes to the VMAP file. Essentially only the queuing of the writes remains to be taken care of.
   It would be typical to conduct the mapping of the suction side and of the pressure side of the blade in parallel. More advanced splits into separate mapping domains may be beneficial, e.g., span-wise segments, the blade tip, or cooling channels in the case of turbine rotors.
3) **Selective storage of results, filtering**
   CFD simulations on HPC systems, for example, may output massive amounts of data. Especially iterative simulation processes with intermediate user input (e.g. in order to reach convergence) may lead to the creation of multiple large CGNS files. Likewise, CSM simulations of higher fidelity produce significant amounts of data that our processes store in VMAP files. The fact that HDF5 files are easy to be filtered allows for easy development of selective storage routines. For instance, the user may select results from a hierarchy of physical domains for archiving, such as specific nodes, single parts or whole assemblies of certain iterations. These possibilities effectively lower the barrier for efficient data management, a topic that is becoming more relevant as the amount of available data increases.
4) **General user and developer comfort**
   General convenience for users and developers is a factor that should not be underestimated. The easy practical usage of HDF5 and the sophisticated interfaces to many popular programming languages are key to the broad adoption of this binary format in general. Combined with the efficiency and performance of storage, HDF5 is

superior to non-binary structured data formats, such as XML, when the amount of data is large. In the view of the authors, there is no reason against the utilization of HDF5 even for comparatively small amounts of data. The likelihood for specialized software and processes to be accepted and furthered by a community is most certainly increased if the produced data are easily usable and manageable regardless of the size of the data. Therefore, we develop our processes with deep HDF5 integration and argue in support of VMAP as a central format for Finite Element data.

B. Finite Element (FE) software interoperability enabling further process automation

In addition to automation of simulation workflows that are specific to engine design, more general scenarios exist in which the employment of HDF5 is a promising source of efficiency gains. These overarching scenarios fall into the categories of organizational collaboration (Subsection V.B.1), interdisciplinary collaboration (Subsection V.B.2), and quality assurance (Subsection V.B.3).

1. Different project stakeholders use different FE software

This is a rather common case, especially in collaborative scenarios. Many software, including but not limited to *CalculiX*, *ABAQUS*, *ANSYS*, *PERMAS*, and *NASTRAN* are capable of performing all standard CSM analyses, for engine design and beyond. This does however not mean that the respective FE models are mutually compatible. Hence, when exchanging model data among collaborators, conversions are often necessary. VMAP and its associated software tools offer capabilities to convert to and from the specific data formats of all of these solvers, cf. [7, "Tools"] and [18]. In this sense, it is well suited to serve as a central FE data format. In the view of the authors, VMAP is the most promising candidate to fulfill this purpose and thus of becoming a CGNS equivalent in the field of CSM.

This is also true when considering pre- and post-processing software. Many of them come along with a wide range of interfaces to different file formats. However, it is important to note that even standard structural analysis of turbo components requires very specific constraints and kinds of analysis, such as cyclic boundary conditions and nodal diameter analyses. Such requirements oftentimes exceed the capabilities of standard processing software. Moreover, the most capable software are commercial and come along with possibly significant recurring costs. The VMAP standard opens possibilities to alternatives such as self-implementation of community-development of specific pre- and post-processing routines that may be shared among collaborators.

2. Consideration of non-standard phenomena

Easy conversion to and from specialized simulation software is an important factor for the choice of the storage format of data. The following list exemplifies advanced scenarios and classes of materials that are important for gas turbine engines. Non-standard numerical methods and sophisticated modelling capabilities are required for the simulation of such problems. It is common to use specific, dedicated software in certain cases. HDF5 lends itself as a basis for such cross-discipline simulations because all required data are readily findable, accessible, and convertible.

- Non-linear friction (e.g. for contact areas of blades and disks, bolts and shrouds)
- Non-linear vibration damping (e.g. for shrouded low-pressure turbine blades)
- Life cycle analysis (e.g. of weld seams, including thermal effects)
- Crack modelling (e.g. by XFEM [19])
- Corrosion
- Erosion
- Impact (cf. FOD)

3. Testing and validation of processes

Automated testing is standard for quality assurance of software. The simulation processes described in Section V.A.3 are regularly tested via Continuous Integration pipelines. By using HDF5 as a standard format, end-to-end tests are readily implemented and maintained.

This is also an important factor when changing the simulation software used for standard or advanced processes. The vendor-agnostic HDF5 format makes the transition from one software to another comparatively easy for process engineers.

C. Memory efficiency

Besides gains in workforce efficiency through automation of turbo component CSM (Section V.A) and of more general CSM (Section V.B), there are also increases in memory efficiency to be achieved by means of the employment of the HDF5 file format.

1. Memory space

Quantitatively, the choice of the storage format directly affects the occupied storage capacity. In any format, certain fundamental decisions must be made in respect thereof. Such decisions include but are not limited to the accuracy of numerical values (e.g. single or double precision) and encoding methods.

HDF5 supports the adjustment of both the precision of numerical values and the encoding of strings (ASCII and UTF-8). In addition, it comes along with native support of compression libraries. Most of these features are usually, e.g. currently within the SMM, set by the automatic initial file-creating process and not a concern of the end-user. However, they may also be made directly accessible to the user, e.g. for advanced simulation engineering and data management.

2. Memory speed

Binary file formats usually outperform text-based formats when it comes to the speed of file input and output. This is mostly due to the fact that, for numerical values, the conversion from or to strings can be omitted. But also line breaks, empty characters and separation characters need not be parsed or written.

On the other hand, this advantage of binary file formats, such as HDF5, over text file formats, such as ASCII, may be partially offset by the introduction of data compression. The latter usually comes along with a high CPU workload that adds to the time necessary for file I/O. Both the speed advantage of bare HDF5 and the speed disadvantage of compression algorithms scale with the size of the involved data.

However, the computational cost of de-/compression should be seen as an additional *option* to the developer or user of the processes. This gives rise to the possibility of prioritizing memory space over memory speed, or vice versa. In this sense, the entirety of processes and their data may be well-tailored to the specific needs of the application.

VI. Conclusion

The extension of the central data model of GTlab to HDF5 data has proven very fruitful. The original goal of deep integration of large amounts of structured numerical data into the collaboration platform has clearly been achieved, as confirmed by feedback from the GTlab community. A significant step in the direction of extending GTlab by higher fidelity design capabilities has thus been taken.

However, this work can not be called finished as implementations of some important features are still pending. Such features are the employment of a full-fledged data management system, the adaption of existing processes to special requirements, the implementation of additional processes, and others.

The CSM perspective of this paper has been the main source of feature requests for the HDF5 module development, see Part A [3]. This closely user-driven development has lead to the rise of new concepts and ideas that had not been envisioned originally. Instead, these have occurred rather spontaneously during the short iteration cycles. Especially requirements concerning user friendliness, but also module-developer friendliness, emerged only during actual experimentation.

The result is a ready-to-release Structural Mechanics Module, enriching the GTlab ecosystem and sustainably freeing personal resources. The latter are now being invested into furthering these developments and also into leveraging the now available automated processes for scientific research.

References

[1] Reitenbach, S., Vieweg, M., Becker, R., Hollmann, C., Wolters, F., Schmeink, J., Otten, T., and Siggel, M., "Collaborative Aircraft Engine Preliminary Design using a Virtual Engine Platform, Part A: Architecture and Methodology," *AIAA Scitech 2020 Forum*, American Institute of Aeronautics and Astronautics, 2020. https://doi.org/10.2514/6.2020-0867.

[2] Vieweg, M., Reitenbach, S., Hollmann, C., Schnös, M., Behrendt, T., Krumme, A., Otten, T., and zu Ummeln, R. M., "Collaborative Aircraft Engine Preliminary Design using a Virtual Engine Platform, Part B: Application," *AIAA Scitech 2020 Forum*, American Institute of Aeronautics and Astronautics, 2020. https://doi.org/10.2514/6.2020-0124.

[3] Bröcker, M., and Kunc, O., "Integration of large data based on HDF5 in a collaborative and multidisciplinary design environment, Part A: Methodology and Implementation," *AIAA SciTech Forum, submitted for publication, session "Digital Engineering"*, 2024.

[4] Reitenbach, S., Vieweg, M., Hollmann, C., and Becker, R. G., "Usage of Data Provenance Models in Collaborative Multidisciplinary Aero-Engine Design," *Journal of Engineering for Gas Turbines and Power*, Vol. 142, No. 10, 2020. https://doi.org/10.1115/1.4048436.

[5] Wolf, K., "VMAP Standard and the VMAP Standards Community e.V." Fraunhofer SCAI, NAFEMS World Congress, 2021. NWC21-204-b.

[6] Gulati, P., "VMAP Enabling Interoperability in Integrated CAE Simulation Workflows," Fraunhofer SCAI, NAFEMS World Congress, 2021. NWC21-205-b.

[7] "VMAP Standard Specifications," https://vmap-standard.org, 2020. Accessed Dec 1, 2023.

[8] Ritt, S. A., Kunc, O., Forsthofer, N., Schwämmle, M., and Reitenbach, S., "Multi-disciplinary fan assessment with respect to bird strike," *Deutscher Luft- und Raumfahrtkongress 2016*, DGLR, 2023.

[9] Geuzaine, C., and Remacle, J.-F., "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, Vol. 79, No. 11, 2009, pp. 1309–1331. https://doi.org/10.1002/nme.2579.

[10] Forsthofer, N., and Reiber, C., "Structural Mechanic and Aeroelastic Approach for Design and Simulation of CFRP Fan Blades," *Deutscher Luft- und Raumfahrtkongress 2016*, DLR, 2016. URL https://elib.dlr.de/109299/.

[11] Gaun, L., Bestle, D., and Huppertz, A., "Hot-to-Cold CAD Geometry Transformation of Aero Engine Parts Based on B-Spline Morphing," 2014, p. V02BT45A020. https://doi.org/10.1115/GT2014-26683.

[12] Dhondt, G., *The Finite Element Method for Three-Dimensional Thermomechanical Applications*, Wiley, 2004. https://doi.org/10.1002/0470021217.

[13] "TRACE User Guide," http://trace-portal.de/userguide/trace/README.html, 2023. Accessed Nov 14, 2023.

[14] de Boer, A., van der Schoot, M., and Bijl, H., "Mesh deformation based on radial basis function interpolation," *Computers & Structures*, Vol. 85, No. 11-14, 2007, pp. 784–795. https://doi.org/10.1016/j.compstruc.2007.01.013.

[15] Schuff, M., "Coupled Mode Flutter of Turbomachinery Blades," Tech. rep., Technische Universität Berlin, 2023. https://doi.org/10.57676/k1d4-mk17.

[16] Carstens, V., and Belz, J., "Numerical Investigation of Nonlinear Fluid-Structure Interaction in Vibrating Compressor Blades," *Journal of Turbomachinery*, Vol. 123, No. 2, 2000, pp. 402–408. https://doi.org/10.1115/1.1354138.

[17] Allemang, R. J., "The modal assurance criterion–twenty years of use and abuse," *Sound and vibration*, Vol. 37, No. 8, 2003, pp. 14–23.

[18] Barth, N., and Kunc, O., "PermasVmap," , Nov. 2022. https://doi.org/10.5281/zenodo.7360515.

[19] Rege, K., and Lemu, H. G., "A review of fatigue crack propagation modelling techniques using FEM and XFEM," *IOP Conference Series: Materials Science and Engineering*, Vol. 276, 2017, p. 012027. https://doi.org/10.1088/1757-899x/276/1/012027.